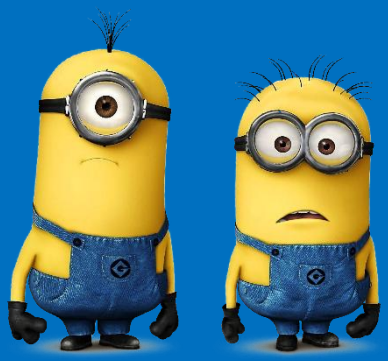
Programming Languages

Лабораторная работа # 5 (15)

Python и ООП. Отношение (взаимосвязь) между классами. Ассоциации и её разновидности: агрегация, композиция и наследование



 Department: <Software of Computer Technology & Computer-Aided Systems>
Faculty of Information Technology and Robotics
Belarusian National Technical University
by Viktor Ivanchenko

ЛАБОРАТОРНАЯ РАБОТА # 5 (15)
Python и ООП. Отношение (взаимо-связь) между классами. Ассоциации и её разновидности: агрегация, композиция и наследование

Цель работы

Изучить механизмы и способы взаимодействия классов и объектов данных классов между собой с использованием разновидностей ассоциации (наследования, агрегации и композиции) и закрепить их на примере проектирования и реализации ООП-программ с использованием языка программирования Python.

Основное задание

Спроектировать и реализовать программную систему, которая бы определяла и реализовывала иерархию различных объектов автотранспорта: легковые машины, микроавтобусы, автобусы, грузовые машины, мини-фургоны и т.д., группировала бы все объекты в объекте автопарка и высчитывала бы следующие значения:

- общее количество пассажиров, которых одновременно может перевезти пассажирский транспорт автопарк;
- общий вес груза, который могут взять одновременно для перевозки все грузовые машины автопарк;
- общее количество топлива, необходимое для заправки всего автопарка машин.

Индивидуальное задание

Произвести рефакторинг программной системы, созданной в предыдущей лабораторной работе, следующим образом:

- ✓ классы, описывающие объекты соответствующей предметной области (бизнес объекты), должны быть сведены в иерархическую структуру (произвести, где это необходимо, классификацию типов);
- ✓ скрыть наборы объектов, которыми манипулирует логика системы, в соответствующие контейнерные классы; логика системы должны принимать на вход объекты только данных контейнерных классов;
- ✓ логика системы должна зависеть только от абстракции, а не от реализации;
- ✓ логика системы должна быть реализована внутри соответствующих функциональных классов (к примеру, в виде статических полей), а не в виде отдельных Python-функций.
- ✓ добавить в методы бизнес логики проверку входящих объектов на соответствие типа, с которым должна взаимодействовать логика.

Требования к выполнению

1) Программа должна обязательно быть снабжена комментариями на английском языке, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика, номер группы и дату разработки.
2) Исходный текст классов и демонстрационной программы рекомендуется снабжать комментариями.
3) В отчёте ОБЯЗАТЕЛЬНО привести UML-диаграмму классов, которая демонстрирует классы и объекты приложения, их атрибуты и методы, а также взаимосвязь между ними.
4) Любые типы отношений между проектируемыми классами (ассоциация, агрегация, композиция, наследование и т.д.) должны применяться лишь тогда, когда это имеет смысл.
5) Каждый класс должен иметь адекватное осмысленное имя (обычно это *имя существительное*) и начинаться с заглавной буквы. Имена полей и методов должны начинаться с маленькой буквы и быть также осмысленными (имя метода, который что-то вычисляет, обычно называют *глаголом*, а поле – именем существительным).
6) При проектировании классов необходимо придерживаться принципа единственной ответственности (Single Responsibility Principle), т.е. классы должны

проектироваться и реализовываться таким образом, чтобы они были менее завязаны с другими классами при своей работе – они должны быть самодостаточными.

7) Каждый класс необходимо разместить в отдельном модуле, который затем подключается в другом модуле, где происходит создание объекта данного класса и его использование.
8) Программа для демонстрации работоспособности разработанных классов должна быть снабжена дружелюбным и интуитивно понятным интерфейсом.
9) При разработки кода необходимо придерживаться соответствующего стиля (соглашения по форматированию и именованию), который используется для языка программирования Python.

Best of LUCK with it, and remember to HAVE FUN while you're learning :)

Victor Ivanchenko